

# Dynamic beats fixed: On phase-based algorithms for file migration\*

Marcin Bienkowski<sup>1</sup>, Jarosław Byrka<sup>1</sup>, and Marcin Mucha<sup>2</sup>

<sup>1</sup> Institute of Computer Science, University of Wrocław, Poland

<sup>2</sup> Institute of Informatics, University of Warsaw, Poland

---

## Abstract

In this paper, we construct a deterministic 4-competitive algorithm for the online file migration problem, beating the currently best 20-year old, 4.086-competitive MTLM algorithm by Bartal et al. (SODA 1997). Like MTLM, our algorithm also operates in phases, but it adapts their lengths dynamically depending on the geometry of requests seen so far. The improvement was obtained by carefully analyzing a linear model (factor-revealing LP) of a single phase of the algorithm. We also show that if an online algorithm operates in phases of fixed length and the adversary is able to modify the graph between phases, no algorithm can beat the competitive ratio of 4.086.

**1998 ACM Subject Classification** F.1.2 Modes of Computation: Online computation, G.1.6 Optimization: Linear programming, F.2.2 Nonnumerical Algorithms and Problems

**Keywords and phrases** file migration, factor-revealing linear programs, online algorithms, competitive analysis

## 1 Introduction

Consider the problem of managing a shared data item among sets of processors. For example, in a distributed program running in a network, nodes want to have access to shared files, objects or databases. Such a file can be stored in the local memory of one of the processors and when another processor wants to access (read from or write to) this file, it has to contact the processor holding the file. Such a transaction incurs a certain cost. Moreover, access patterns to this file may change frequently and unpredictably, which renders any static placement of the file inefficient. Hence, the goal is to minimize the total cost of communication by moving the file in response to such accesses, so that the requesting processors find the file “nearby” in the network.

The *file migration* problem serves as the theoretical underpinning of the application scenario described above. The problem was coined by Black and Sleator [13] and was initially called *page migration*, as the original motivation concerned managing a set of memory pages in a multiprocessor system. There the data item was a single memory page held at a local memory of a single processor.

Most subsequent work referred to this problem as *file migration* and we will stick to this convention in this paper. The file migration problem assumes the *non-uniform model*, where the shared file is much larger than a portion accessed in a single time step. This is typical when in one step a processor wants to read a single unit of data from a file or a record from




---

\* Partially supported by Polish National Science Centre grants 2016/22/E/ST6/00499 and 2015/18/E/ST6/00456. The work of M. Mucha is part of a project TOTAL that has received funding from the European Research Council (ERC) under the European Union’s Horizon 2020 research and innovation programme (grant agreement No 677651).



© Marcin Bienkowski, Jarosław Byrka and Marcin Mucha;  
licensed under Creative Commons License CC-BY



Leibniz International Proceedings in Informatics  
LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

a database. On the other hand, to reduce the maintenance overhead, it is assumed that the shared file is indivisible, and can be migrated between nodes only as a whole. This makes the file migration much more expensive than a single access to the file. As the knowledge of future accesses is either partial or completely non-existing, the accesses to the file can be naturally modeled as an online problem, where the input sequence consists of processor identifiers, which sequentially try to access pieces of the shared file.

## 1.1 The Model

The studied network is modeled as an edge-weighted graph or, more generally, as a metric space  $(\mathcal{X}, d)$  whose point set  $\mathcal{X}$  corresponds to processors and  $d$  defines the distances between them. There is a large indivisible *file* (historically called *page*) of size  $D$  stored at a point of  $\mathcal{X}$ . An input is a sequence of space points  $r_1, r_2, r_3, \dots$  denoting processors requesting access to the file. This sequence is presented in an online manner to an algorithm. More precisely, we assume that the time is slotted into steps numbered from 1. Let  $\text{ALG}_t$  denote the position of the file at the end of step  $t$  and  $\text{ALG}_0$  be the initial position of the file. In step  $t \geq 1$ , the following happens:

1. A requesting point  $r_t$  is presented to the algorithm.
2. The algorithm pays  $d(\text{ALG}_{t-1}, r_t)$  for serving the request.
3. The algorithm chooses a new position  $\text{ALG}_t$  for the file (possibly  $\text{ALG}_t = \text{ALG}_{t-1}$ ) and moves the file to  $\text{ALG}_t$  paying  $D \cdot d(\text{ALG}_{t-1}, \text{ALG}_t)$ .

After the  $t$ -th request, the algorithm has to make its decision (where to migrate the file) exclusively on the basis of the sequence up to step  $t$ . To measure the performance of an online strategy, we use the standard competitive ratio metric [14]: an online deterministic algorithm  $\text{ALG}$  is  $c$ -competitive if there exists a constant  $\gamma$ , such that for any input sequence  $\mathcal{I}$ , it holds that  $C_{\text{ALG}}(\mathcal{I}) \leq c \cdot C_{\text{OPT}}(\mathcal{I}) + \gamma$ , where  $C_{\text{ALG}}$  and  $C_{\text{OPT}}$  denote the costs of  $\text{ALG}$  and  $\text{OPT}$  (optimal *offline* algorithm) on  $\mathcal{I}$ , respectively. The minimum  $c$  for which  $\text{ALG}$  is  $c$ -competitive is called the *competitive ratio* of  $\text{ALG}$ .

## 1.2 Previous Work

The problem was stated by Black and Sleator [13], who gave 3-competitive deterministic algorithms for uniform metrics and trees and conjectured that 3-competitive deterministic algorithms were possible for any metric space.

Westbrook [26] constructed randomized strategies: a 3-competitive algorithm against adaptive-online adversaries and a  $(1 + \phi)$ -competitive algorithm (for  $D$  tending to infinity) against oblivious adversaries, where  $\phi \approx 1.618$  denotes the golden ratio. By the result of Ben-David et al. [10] this asserted the *existence* of a deterministic algorithm with the competitive ratio at most  $3 \cdot (1 + \phi) \approx 7.854$ .

The first explicit deterministic construction was the 7-competitive algorithm MOVE-TO-MIN (MTM) by Awerbuch et al. [2]. MTM operates in phases of length  $D$ , during which the algorithm *remains at a fixed position*. In the last step of a phase, MTM migrates the file to a point that minimizes the sum of distances to all requests  $r_1, r_2, \dots, r_D$  presented in the phase, i.e., to a minimizer of the function  $f_{\text{MTM}}(x) = \sum_{i=1}^D d(x, r_i)$ .

The ratio has been subsequently improved by the algorithm MOVE-TO-LOCAL-MIN (MTLM) by Bartal et al. [8]. MTLM works similarly to MTM, but it changes the phase duration to  $c_0 \cdot D$  for a constant  $c_0$ , and when computing the new position for the file, it

also takes the migration distance into consideration. Namely, it chooses to migrate the file to a point that minimizes the function

$$f_{\text{MTLM}}(x) = D \cdot d(v_{\text{MTLM}}, x) + \frac{c_0+1}{c_0} \sum_{i=1}^{c_0 \cdot D} d(x, r_i) ,$$

where  $v_{\text{MTLM}}$  denotes the point at which MTLM keeps its file during the phase. The algorithm is optimized by setting  $c_0 \approx 1.841$  being the only positive root of the equation  $3c^3 - 8c - 4 = 0$ . For such  $c$ , the competitive ratio of MTLM is  $R_0 \approx 4.086$ , where  $R_0$  is the largest (real) root of the equation  $R^3 - 5R^2 + 3R + 3 = 0$ . Their analysis is tight.

It is worth noting that most of the competitive ratios given above hold when  $D$  tends to infinity. In particular, for MTLM we assume that  $c_0 \cdot D$  is an integer and the ratio of  $1 + \phi$  of Westbrook's algorithm [26] is achieved only in the limit.

Better deterministic algorithms are known only for some specific graph topologies. There are 3-competitive algorithms for uniform metrics and trees [13], and  $(3 + 1/D)$ -competitive strategies for three-point metrics [23]. Chrobak et al. [15] showed  $2 + 1/(2D)$ -competitive strategies for continuous trees and products of trees, e.g., for  $\mathbb{R}^n$  with  $\ell_1$  norm. Furthermore, a  $(1 + \phi)$ -competitive algorithm for  $\mathbb{R}^n$  under any norm was also given in [15].

A straightforward lower bound of 3 for deterministic algorithms was given by Black and Sleator [13] and later adapted to randomized algorithms against adaptive-online adversaries by Westbrook [26]. The currently best lower bound for deterministic algorithms is due to Matsubayashi [22], who showed a lower bound of  $3 + \varepsilon$  that holds for any value of  $D$ , where  $\varepsilon$  is a constant that does not depend on  $D$ . This renders the file migration problem one of the few natural problems, where a known lower bound on the competitive ratio of any deterministic algorithm is strictly larger than the competitive ratio of a randomized algorithm against an adaptive-online adversary.

Finally, improved results were given for a simplified model where  $D = 1$ : the competitive ratio for deterministic algorithms is then known to be between 3.164 and 3.414 [21].

### 1.3 Our Contribution

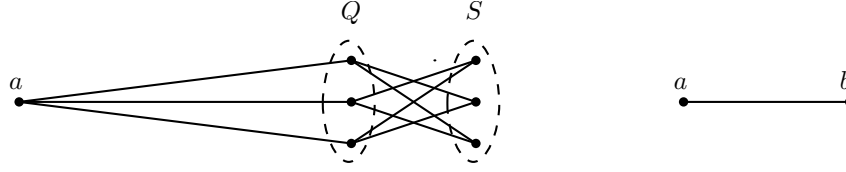
We propose a new deterministic algorithm that dynamically decides on the length of the phase based on the geometry of requests received in the initial part of each phase. This improves the 20-years old result of Bartal et al. [8].

The improvement was obtained by carefully analyzing a linear model (factor revealing LP) of a single phase of the algorithm. It allowed us to identify some key tight examples for the previous analysis, suggested a nontrivial construction of the new algorithm, and facilitated a systematic search within the design parameter space.

More precisely, for a fixed algorithm ALG (from a relatively broad class), we create a maximization LP with the following property: if the competitive ratio of ALG is at least  $R$ , then so is the value of LP. A solution to the LP contains a succinct description of a metric space along with a short description of a single-phase input, both constituting a lower bound for ALG. Hence, the value of LP is an upper bound on ALG's competitiveness. We discuss the details of the LP approach in Section 4.

The way the algorithm was obtained is perhaps unintuitive. Nevertheless, the final algorithm is an elegant construction involving only essentially integral constants. By studying the dual solution, we managed to extract a compact, human-readable, combinatorial upper bound based on path-packing arguments and to obtain the following result.

► **Theorem 1.** *There exists a deterministic 4-competitive algorithm for the file migration problem.*



■ **Figure 1** The geometries of selected tight instances for MTLM. In both cases, the algorithm starts at point  $a$ . In the *linear* instance (on the right), the requests are initially given at  $a$  and then later at  $b$ , and the algorithm is expected to migrate the file to point  $b$ . In the *bipartite* instance (on the left) the requests are given at nodes from set  $S$  and the algorithm is expected to migrate the file to one of the nodes from set  $Q$ .

We also show that improvement of MTLM would not be possible by just selecting different parameters for a phase-based algorithm operating with a fixed phase length. Our construction shows that an analysis that treats each phase separately (e.g., the one employed for MTLM [8]) cannot give better bounds on the competitive ratio than 4.086. (A weaker lower bound of 3.847 for algorithms that use fixed phase length was given by Bartal et al. [8].)

► **Theorem 2.** *Fix any algorithm ALG that operates in phases of fixed length. Assume that between the phases, the adversary can arbitrarily modify the graph while keeping the distance between the files of OPT and ALG unchanged. Then, the competitive ratio of ALG is at least  $R_0$  (for  $D$  tending to infinity), where  $R_0 \approx 4.086$  is the competitive ratio of algorithm MTLM.*

## 1.4 Other Related Work

The file migration problem has been generalized in a few directions. When we lift the restriction that the file can only be migrated and not copied, the resulting problem is called *file allocation* [9, 2, 18]. It makes sense especially when we differentiate read and write requests to the file; for the former, we need to contact only one replica of the file; for the latter all copies need to be updated. The attainable competitive ratios become then worse: the best deterministic algorithm is  $O(\log n)$ -competitive [2]; the lower bound of  $\Omega(\log n)$  holds even for randomized algorithms and follows by a reduction from the online Steiner tree problem [9, 17].

The file migration problem has been also extended to accommodate memory capacity constraints at nodes (when more than one file is used) [1, 3, 4, 6], dynamically changing networks [4, 12], and different objective functions (e.g., minimizing congestion) [19, 25]. For a more systematic treatment of the file migration and related problems, see surveys [7, 11]. For more applied approaches, see the survey [16] and the references therein.

## 2 4-Competitive Algorithm Dynamic-Local-Min

We start with an insight concerning the hard inputs for the MTLM algorithm [8]. We identified two classes of tight instances for MTLM: *bipartite* and *linear* (cf. Figure 1). It can be shown that if the algorithm knew in advance on which instance it was run, it could improve its performance by changing the phase length. Namely, for bipartite instances a longer phase would help the algorithm, whereas a shorter phase would be beneficial for linear instances.

To decide the length of the phase, we need to measure the level of request concentration as compared to the distance from the current position of an algorithm to the center of requests. Intuitively, observing that (from some time) requests are concentrated around a certain

point motivates the algorithm to shorten the phase and quickly move to the “center of the requests”. If, on the other hand, requests are scattered and the current algorithm’s position is essentially in the middle of the observed requests, it appears reasonable to wait longer before moving the file. This rule agrees with the desired behavior of the algorithm on linear and bipartite instances.

Turning the above intuition into an effective phase extension rule is not trivial. We present an algorithm based on a rule that we have extracted from an optimization process using a natural linear model of the amortized phase-based analysis. This linear model is quite complex and we present it in [Section 4](#). It can be seen as an alternative (computer-based) proof for the performance guarantee of our algorithm. Such proof technique might be interesting on its own and useful for analyzing other online games played on metric spaces.

## 2.1 Notation

For succinctness, we introduce the following notions. For any two points  $v_1, v_2 \in \mathcal{X}$ , let  $[v_1, v_2] = D \cdot d(v_1, v_2)$ . We extend this notation to sequences of points, i.e.,  $[v_1, v_2, \dots, v_j] = [v_1, v_2] + [v_2, v_3] + \dots + [v_{j-1}, v_j]$ . Moreover, if  $v \in \mathcal{X}$  is a point and  $S \subseteq \mathcal{X}$  is a multiset of points, then

$$[v, S] = [S, v] = D \cdot \frac{1}{|S|} \sum_{x \in S} d(v, x) ,$$

i.e.,  $[v, S]$  is the average distance from  $v$  to a point of  $S$  times  $D$ . We extend the sequence notation introduced above to sequences of points and multisets of points, e.g.,  $[v, S, u, T] = [v, S] + [S, u] + [u, T]$ . The symbol  $[S, T]$  is not defined for multisets  $S, T$ ; we will only use this notation for sequences that do not contain two consecutive multisets.

Observe that the sequence notation allows for easy expressing of the triangle inequality:  $[v_1, v_2] \leq [v_1, v_3, v_2]$ ; we will extensively use this property. Note that the following “multiset” version of the triangle inequality also holds:  $[v_1, v_2] \leq [v_1, S, v_2]$ .

## 2.2 Algorithm definition

We propose a new phase-based algorithm that dynamically decides on the length of the current phase, which we call Dynamic-Local-Min (DLM). DLM operates in phases, but it chooses their lengths depending on the geometry of requests seen in the initial part of the phase. Roughly speaking, when it recognizes that the currently seen requests more closely resemble a linear tight example for MTLM, it ends the phase after  $1.75 D$  steps. Otherwise, it assumes that the presented graph is more in the flavor of the bipartite construction, and ends the phase only after  $2.25 D$  steps.

For any step  $t$ , we denote the position of DLM’s file at the end of step  $t$  by  $\text{DLM}_t$  and that of OPT by  $\text{OPT}_t$ . We identify the requests with the points where they are issued.

Assume a phase starts in step  $t + 1$ ; that is,  $\text{DLM}_t$  is the position of DLM at the very beginning of a phase. Within the phase, DLM waits  $1.75 D$  steps and at step  $t + 1.75 D$ , it finds a node  $v_g$  that minimizes the function

$$g(v) = [\text{DLM}_t, v, \mathcal{R}_1, v, \mathcal{R}_2] = [\text{DLM}_t, v] + 2 \cdot [v, \mathcal{R}_1] + [v, \mathcal{R}_2] ,$$

where  $\mathcal{R}_1$  is the multiset of the requests from steps  $t + 1, \dots, t + D$  and  $\mathcal{R}_2$  is the multiset of the subsequent requests from steps  $t + D + 1, \dots, t + 1.75 D$ .

If  $g(v_g) \leq 1.5 \cdot [\text{DLM}_t, \mathcal{R}_2]$ , the algorithm moves its file to  $v_g$ , and ends the current phase. Intuitively, this condition corresponds to detecting if there exists a point that is substantially

## A:6 Dynamic beats fixed: On phase-based algorithms for file migration

closer to the first  $1.75D$  requests of the phase than the current position. The condition is constructed in a way to be conclusive for each of the possible outcomes. If indeed such point exists, then by moving the file to this point, we get much closer to the file of OPT. If there is no such good point, then also the optimal solution is experiencing some request related costs. Then, we may afford to wait a little longer and meanwhile get a more accurate estimation of the possible location of the file of OPT.

That is, if  $g(v_g) > 1.5 \cdot [\text{DLM}_t, \mathcal{R}_2]$ , DLM waits the next  $0.5D$  steps and (in step  $t + 2.25D$ ) it moves its file to the point  $v_h$ , where  $v_h$  is the minimizer of the function

$$h(v) = [\text{DLM}_t, v] + [v, \mathcal{R}_1] + 1.25 \cdot [v, \mathcal{R}_2] + 0.75 \cdot [v, \mathcal{R}_3] .$$

$\mathcal{R}_3$  is the multiset of the last  $0.5D$  requests from the prolonged phase (from steps  $t + 1.75D + 1, \dots, t + 2.25D$ ). Also in this case, the next phase starts right after the file movement.

Note that the *short phase* consists of  $D$  requests denoted  $\mathcal{R}_1$  followed by  $0.75D$  requests denoted  $\mathcal{R}_2$ , while the *long phase* consists additionally of  $0.5D$  requests denoted  $\mathcal{R}_3$ . We will say that the short phase consists of *two parts*,  $\mathcal{R}_1$  and  $\mathcal{R}_2$ , and the long phase consists of *three parts*,  $\mathcal{R}_1$ ,  $\mathcal{R}_2$  and  $\mathcal{R}_3$ .

### 2.3 DLM Analysis

We start with a lower bound on OPT. The following bound is an extension of the bound given implicitly in [8]; we present its proof for completeness in [Appendix A](#).

► **Lemma 3.** *Let  $\mathcal{R}$  be a subsequence of at most  $2D$  consecutive requests from the input issued at steps  $t + 1, t + 2, \dots, t + |\mathcal{R}|$ . Then,  $4 \cdot C_{\text{OPT}}(\mathcal{R}) \geq (2|\mathcal{R}|/D) \cdot [\text{OP}_t, \mathcal{R}, \text{OP}_{t+|\mathcal{R}|}] + (4 - 2|\mathcal{R}|/D) \cdot [\text{OP}_t, \text{OP}_{t+|\mathcal{R}|}]$ .*

We define a potential function at (the end of) step  $t$  as  $\Phi_t = 3 \cdot [\text{DLM}_t, \text{OP}_t]$ . It suffices to show that in any (short or long) phase consisting of steps  $t + 1, t + 2, \dots, t + \ell$ , during which requests  $\mathcal{R}$  are given, it holds that

$$C_{\text{ALG}}(\mathcal{R}) + \Phi_{t+\ell} \leq 4 \cdot C_{\text{OPT}}(\mathcal{R}) + \Phi_t . \quad (1)$$

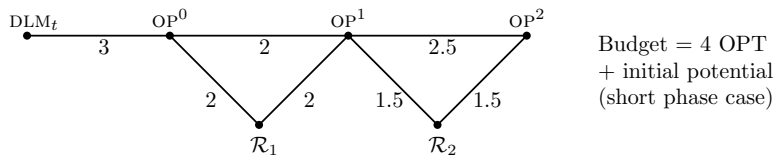
[Theorem 1](#) follows immediately by summing the above bound over all phases of the input.

#### 2.3.1 Proof for a short phase.

We consider any short phase  $\mathcal{R}$  consisting of part  $\mathcal{R}_1$ , spanning steps  $t + 1, \dots, t + D$ , and part  $\mathcal{R}_2$ , spanning steps  $t + D + 1, \dots, t + 1.75D$ . For succinctness, we define  $\text{OP}^0 = \text{OP}_t$ ,  $\text{OP}^1 = \text{OP}_{t+D}$  and  $\text{OP}^2 = \text{OP}_{t+1.75D}$ . By [Lemma 3](#) applied to  $\mathcal{R}_1$  and  $\mathcal{R}_2$ ,

$$\begin{aligned} 4 \cdot C_{\text{OPT}}(\mathcal{R}) + \Phi_t &= 3 \cdot [\text{DLM}_t, \text{OP}^0] + 4 \cdot C_{\text{OPT}}(\mathcal{R}_1) + 4 \cdot C_{\text{OPT}}(\mathcal{R}_2) \\ &\geq 3 \cdot [\text{DLM}_t, \text{OP}^0] + 2 \cdot [\text{OP}^0, \text{OP}^1] + 2 \cdot [\text{OP}^0, \mathcal{R}_1, \text{OP}^1] \\ &\quad + 2.5 \cdot [\text{OP}^1, \text{OP}^2] + 1.5 \cdot [\text{OP}^1, \mathcal{R}_2, \text{OP}^2] . \end{aligned} \quad (2)$$

We treat the amount (2) as our budget. This is illustrated below; the coefficients are written as edge weights.



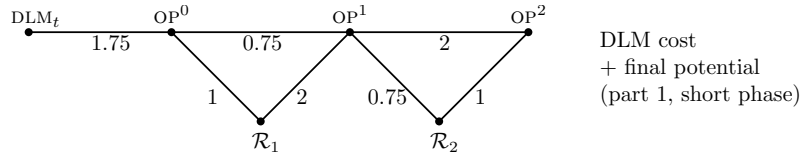
Now, we bound  $C_{\text{ALG}}(\mathcal{R}) + \Phi_{t+1.75D}$  using the definition of ALG and the triangle inequality.

$$\begin{aligned}
C_{\text{ALG}}(\mathcal{R}) + \Phi_{t+1.75D} &= C_{\text{ALG}}(\mathcal{R}_1) + C_{\text{ALG}}(\mathcal{R}_2) + 3[v_g, \text{OP}^2] \\
&\leq [\text{DLM}_t, \mathcal{R}_1] + 0.75 \cdot [\text{DLM}_t, \mathcal{R}_2] + [\text{DLM}_t, v_g] + 3 \cdot [v_g, \text{OP}^2] \\
&\leq [\text{DLM}_t, \mathcal{R}_1] + 0.75 \cdot [\text{DLM}_t, \mathcal{R}_2] + [\text{DLM}_t, v_g] + 2 \cdot [v_g, \mathcal{R}_1, \text{OP}^2] + [v_g, \mathcal{R}_2, \text{OP}^2] \\
&= [\text{DLM}_t, \mathcal{R}_1] + 0.75 \cdot [\text{DLM}_t, \mathcal{R}_2] + 2 \cdot [\text{OP}^2, \mathcal{R}_1] + [\text{OP}^2, \mathcal{R}_2] \\
&\quad + [\text{DLM}_t, v_g] + 2 \cdot [v_g, \mathcal{R}_1] + [v_g, \mathcal{R}_2] \\
&= [\text{DLM}_t, \mathcal{R}_1] + 0.75 \cdot [\text{DLM}_t, \mathcal{R}_2] + 2 \cdot [\text{OP}^2, \mathcal{R}_1] + [\text{OP}^2, \mathcal{R}_2] + g(v_g) . \tag{3}
\end{aligned}$$

The first four summands of (3) can be bounded as

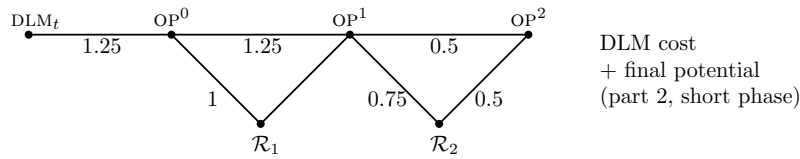
$$\begin{aligned}
&[\text{DLM}_t, \mathcal{R}_1] + 0.75 \cdot [\text{DLM}_t, \mathcal{R}_2] + 2 \cdot [\text{OP}^2, \mathcal{R}_1] + [\text{OP}^2, \mathcal{R}_2] \\
&\leq [\text{DLM}_t, \text{OP}^0, \mathcal{R}_1] + 0.75 \cdot [\text{DLM}_t, \text{OP}^0, \text{OP}^1, \mathcal{R}_2] + 2 \cdot [\text{OP}^2, \text{OP}^1, \mathcal{R}_1] + [\text{OP}^2, \mathcal{R}_2] , \tag{4}
\end{aligned}$$

and their total weights in the final expression are depicted below.



For the last summand of (3),  $g(v_g)$ , we use the fact that  $v_g$  is a minimizer of the function  $g$  (and hence  $g(v_g) \leq g(\text{OP}^0)$ ), and the property of the short phase ( $g(v_g) \leq 1.5 \cdot [\text{DLM}_t, \mathcal{R}_2]$ ). Therefore,

$$\begin{aligned}
g(v_g) &\leq 0.5 \cdot g(\text{OP}^0) + 0.75 \cdot [\text{DLM}_t, \mathcal{R}_2] \\
&\leq 0.5 \cdot [\text{DLM}_t, \text{OP}^0, \mathcal{R}_1, \text{OP}^0, \mathcal{R}_2] + 0.75 \cdot [\text{DLM}_t, \mathcal{R}_2] \\
&\leq 0.5 \cdot [\text{DLM}_t, \text{OP}^0, \mathcal{R}_1, \text{OP}^0, \text{OP}^1, \text{OP}^2, \mathcal{R}_2] + 0.75 \cdot [\text{DLM}_t, \text{OP}^0, \text{OP}^1, \mathcal{R}_2] . \tag{5}
\end{aligned}$$



By combining (3), (4) and (5) (or simply adding the edge coefficients on the last two figures) we observe that the budget ((2), i.e., the edge coefficients on the first figure) is not exceeded. This implies 4-competitiveness, i.e., that (1) holds for any short phase.

### 2.3.2 Proof for a long phase.

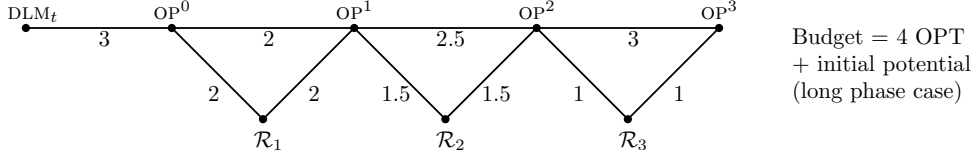
We consider any long phase  $\mathcal{R}$  consisting of part  $\mathcal{R}_1$ , spanning steps  $t+1, \dots, t+D$ ; part  $\mathcal{R}_2$ , spanning steps  $t+D+1, \dots, t+1.75 \cdot D$ ; and part  $\mathcal{R}_3$ , spanning steps  $t+1.75 \cdot D+1, \dots, t+2.25 \cdot D$ . Similarly to the proof for a short phase, we define  $\text{OP}^0 = \text{OP}_t$ ,  $\text{OP}^1 = \text{OP}_{t+D}$ ,  $\text{OP}^2 = \text{OP}_{t+1.75D}$ , and  $\text{OP}^3 = \text{OP}_{t+2.25D}$ . We emphasize that the positions of OPT in a long and a short phase can be completely different.



## A:8 Dynamic beats fixed: On phase-based algorithms for file migration

By [Lemma 3](#), we obtain a bound very similar to that for a short phase; again, we treat it as a budget and depict its coefficients as edge weights.

$$\begin{aligned}
 4 \cdot C_{\text{OPT}}(\mathcal{R}) + \Phi_t &= 3 \cdot [\text{DLM}_t, \text{OP}^0] + 4 \cdot C_{\text{OPT}}(\mathcal{R}_1) + 4 \cdot C_{\text{OPT}}(\mathcal{R}_2) + 4 \cdot C_{\text{OPT}}(\mathcal{R}_3) \\
 &\geq 3 \cdot [\text{DLM}_t, \text{OP}^0] + 2 \cdot [\text{OP}^0, \text{OP}^1] + 2 \cdot [\text{OP}^0, \mathcal{R}_1, \text{OP}^1] \\
 &\quad + 2.5 \cdot [\text{OP}^1, \text{OP}^2] + 1.5 \cdot [\text{OP}^1, \mathcal{R}_2, \text{OP}^2] \\
 &\quad + 3 \cdot [\text{OP}^2, \text{OP}^3] + [\text{OP}^2, \mathcal{R}_3, \text{OP}^3] .
 \end{aligned} \tag{6}$$

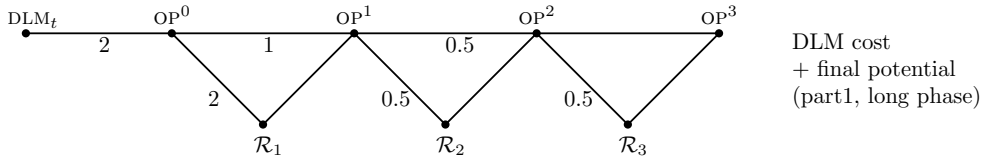


Now, we bound  $C_{\text{ALG}}(\mathcal{R}) + \Phi_{t+2.25D}$ , using the definition of ALG and the triangle inequality.

$$\begin{aligned}
 C_{\text{ALG}}(\mathcal{R}) + \Phi_{t+2.25D} &= C_{\text{ALG}}(\mathcal{R}_1) + C_{\text{ALG}}(\mathcal{R}_2) + C_{\text{ALG}}(\mathcal{R}_3) + 3 \cdot [v_h, \text{OP}^3] \\
 &= [\text{DLM}_t, \mathcal{R}_1] + 0.75 \cdot [\text{DLM}_t, \mathcal{R}_2] + 0.5 \cdot [\text{DLM}_t, \mathcal{R}_3] + [\text{DLM}_t, v_h] + 3 \cdot [v_h, \text{OP}^3] \\
 &\leq [\text{DLM}_t, \mathcal{R}_1] + 0.75 \cdot [\text{DLM}_t, \mathcal{R}_2] + 0.5 \cdot [\text{DLM}_t, \mathcal{R}_3] + [\text{DLM}_t, v_h] \\
 &\quad + [v_h, \mathcal{R}_1, \text{OP}^3] + 1.25 \cdot [v_h, \mathcal{R}_2, \text{OP}^3] + 0.75 \cdot [v_h, \mathcal{R}_3, \text{OP}^3] \\
 &= [\text{DLM}_t, \mathcal{R}_1] + 0.75 \cdot [\text{DLM}_t, \mathcal{R}_2] + 0.5 \cdot [\text{DLM}_t, \mathcal{R}_3] \\
 &\quad + [\text{OP}^3, \mathcal{R}_1] + 1.25 \cdot [\text{OP}^3, \mathcal{R}_2] + 0.75 \cdot [\text{OP}^3, \mathcal{R}_3] + h(v_h) .
 \end{aligned} \tag{7}$$

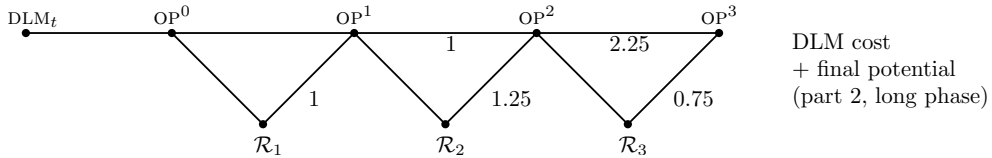
Since DLM has not migrated the file after the first two parts,  $g(v) \geq 1.5 \cdot [\text{DLM}_t, \mathcal{R}_2]$  for any node  $v$ . Therefore  $0.75 \cdot [\text{DLM}_t, \mathcal{R}_2] \leq 0.5 \cdot g(\text{OP}^0) = 0.5 \cdot [\text{DLM}_t, \text{OP}^0, \mathcal{R}_1, \text{OP}^0, \mathcal{R}_2] \leq 0.5 \cdot [\text{DLM}_t, \text{OP}^0, \mathcal{R}_1, \text{OP}^0, \text{OP}^1, \mathcal{R}_2]$ . Using this and the triangle inequality, the first three summands of (7) can be bounded and depicted as follows:

$$\begin{aligned}
 &[\text{DLM}_t, \mathcal{R}_1] + 0.75 \cdot [\text{DLM}_t, \mathcal{R}_2] + 0.5 \cdot [\text{DLM}_t, \mathcal{R}_3] \\
 &\leq [\text{DLM}_t, \text{OP}^0, \mathcal{R}_1] + 0.5 \cdot [\text{DLM}_t, \text{OP}^0, \mathcal{R}_1, \text{OP}^0, \text{OP}^1, \mathcal{R}_2] \\
 &\quad + 0.5 \cdot [\text{DLM}_t, \text{OP}^0, \text{OP}^1, \text{OP}^2, \mathcal{R}_3] .
 \end{aligned} \tag{8}$$



The next three summands of (7) can be also bounded appropriately:

$$\begin{aligned}
 &[\text{OP}^3, \mathcal{R}_1] + 1.25 \cdot [\text{OP}^3, \mathcal{R}_2] + 0.75 \cdot [\text{OP}^3, \mathcal{R}_3] \\
 &\leq [\text{OP}^3, \text{OP}^2, \text{OP}^1, \mathcal{R}_1] + 1.25 \cdot [\text{OP}^3, \text{OP}^2, \mathcal{R}_2] + 0.75 \cdot [\text{OP}^3, \mathcal{R}_3] .
 \end{aligned} \tag{9}$$

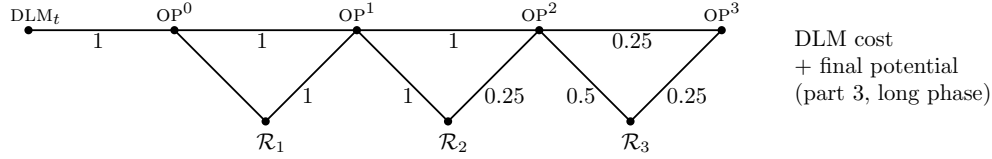




Lastly, for bounding  $h(v_h)$ , we use the fact that  $v_h$  is a minimizer of  $h$ , and hence

$$\begin{aligned}
 h(v_h) &\leq h(\text{OP}^1) \\
 &= [\text{OP}^1, \text{DLM}_t] + [\text{OP}^1, \mathcal{R}_1] + 1.25 \cdot [\text{OP}^1, \mathcal{R}_2] + 0.75 \cdot [\text{OP}^1, \mathcal{R}_3] \\
 &\leq [\text{OP}^1, \text{OP}^0, \text{DLM}_t] + [\text{OP}^1, \mathcal{R}_1] + [\text{OP}^1, \mathcal{R}_2] + 0.25 \cdot [\text{OP}^1, \text{OP}^2, \mathcal{R}_2] \\
 &\quad + 0.5 \cdot [\text{OP}^1, \text{OP}^2, \mathcal{R}_3] + 0.25 \cdot [\text{OP}^1, \text{OP}^2, \text{OP}^3, \mathcal{R}_3] .
 \end{aligned} \tag{10}$$

Note that in (10) we split some of the paths and choose the longer ones, so that the budgets on edges are not violated. Bound (10) is depicted on the figure below.



By combining (7), (8), (9) and (10) (or simply adding edge coefficients on the last three figures), we observe that the budget ((6), i.e., the edge coefficients on the first figure) is not exceeded. This implies 4-competitiveness, i.e., that (1) holds for any long phase and concludes the proof of Theorem 1.

### 3 Lower Bound for Phase-Based Algorithms

A *fixed-phase algorithm* chooses phase length  $c \cdot D$  and after every  $c \cdot D$  requests it makes a migration decision solely on the basis of its current position and the last  $c \cdot D$  requests. We now proceed to argue that no fixed-phase algorithm ALG can beat the competitive ratio  $R_0 \approx 4.086$  achieved by MTLM [8] (cf. Theorem 2). As already stated in the introduction, to ensure that the algorithm cannot base its choices on the previous phases, we will give the adversary an additional power: it may modify the graph between the phases of ALG, as long as the distance between nodes keeping the files of ALG and OPT ( $v_{\text{ALG}}$  and  $v_{\text{OPT}}$ , respectively) is preserved. We emphasize that the analysis of MTLM [8] essentially uses this model: each phase is analyzed completely separately from others. The full proof is presented in Appendix B; here we informally highlight its key ideas.

The adversarial construction consists of many epochs, each consisting of some number of phases. At the beginning and at the end of an epoch, ALG and OPT keep their files at the same node. We define three adversarial strategies, called *plays*: *linear*, *bipartite*, and *finishing*. Each play consists of one or more phases. A prerequisite for each given play is a particular distance between  $v_{\text{ALG}}$  and  $v_{\text{OPT}}$ . Each play will have some properties: it will incur some cost on ALG and OPT and will end with  $v_{\text{ALG}}$  and  $v_{\text{OPT}}$  in a specific distance.

In the first phase of an epoch, when initially  $v_{\text{ALG}} = v_{\text{OPT}}$ , the adversary uses the *linear* play (the generated graph is a single edge of length 1), so that at the end of the phase,  $d(v_{\text{ALG}}, v_{\text{OPT}}) = 1$ . For such phase  $P$ , we have  $C_{\text{ALG}}(P) \geq R_0 \cdot C_{\text{OPT}}(P) - (1/(1 - 2\alpha)) \cdot D$ , where  $\alpha = 1/(R_0 - 1)$ . Note that in this phase alone, the adversary does not enforce the desired competitive ratio of  $R_0$ , but it increases the distance between  $v_{\text{OPT}}$  and  $v_{\text{ALG}}$ .

In each of the next  $L$  phases, the adversary employs the *bipartite* play; the graph used corresponds to a tight bipartite example for MTLM, cf. Figure 1). Let  $f$  be the value of  $d(v_{\text{ALG}}, v_{\text{OPT}})$  at the beginning of a phase. If the algorithm plays well, then at the end of the phase this distance decreases to  $2\alpha \cdot f$ . Furthermore, neglecting lower order terms, for such phase  $P$ , it holds that  $C_{\text{ALG}}(P) \geq R_0 \cdot C_{\text{OPT}}(P) + f \cdot D$ , i.e., the inequality  $C_{\text{ALG}}(P) \geq R_0 \cdot C_{\text{OPT}}(P)$  holds with the slack  $f \cdot D$ . The sum of these slacks over  $L$  phases

is  $\sum_{i=0}^{L-1} (2\alpha)^i \cdot D$ , which tends to  $(1/(1-2\alpha)) \cdot D$  when  $L$  grows. Hence, after one linear and  $L$  bipartite phases (for a large  $L$  and neglecting lower order terms again), the cost paid by ALG is at least  $R_0$  times the cost paid by OPT and the distance between their files is negligible.

Finally, to decrease the distance between  $v_{\text{ALG}}$  and  $v_{\text{OPT}}$  to zero, the adversary uses a third type of play, the *finishing* one. This play incurs a negligible cost and it forces the positions of ALG and OPT files to coincide, which ends an epoch.

## 4 Linear Program for File Migration

In this section, we present a linear programming model for the analysis of both algorithm MTLM by Bartal et al. [8] and our algorithm DLM.

### 4.1 LP analysis of MTLM-like algorithms

In our approach, we analyze any MTLM-like algorithm ALG. We use our notion of distances from Section 2.1. ALG will be a variant of MTLM parameterized with two values  $\beta$  and  $\delta$ . The length of its phase is  $\delta \cdot D$  and the initial point of ALG is denoted by  $A_0$ . We denote the set of requests within a phase by  $\mathcal{R}$ . At the end of a phase, ALG migrates the file to a point  $A_1$  that minimizes the function

$$f(x) = [A_0, x] + \beta \cdot [x, \mathcal{R}] .$$

As in the amortized analysis of the algorithm MTLM [8], we will use a potential function equal to  $\phi$  times the distance between the files of ALG and OPT, where  $\phi$  is a parameter used in the analysis. We let  $O_0$  and  $O_1$  denote the initial and final position of OPT during the studied phase, respectively. Then, the amortized cost of ALG in a single phase is  $C_{\text{ALG}} = \delta \cdot [A_0, \mathcal{R}] + [A_0, A_1] + \phi([A_1, O_1] - [A_0, O_0])$ .

The following factor-revealing LP mimics the proof given in [8]. Namely, it encodes inequalities that are true for any phase and a graph on which ALG can be run. Its goal is to maximize the ratio between  $C_{\text{ALG}}$  and  $C_{\text{OPT}}$ : as an instance can be scaled, we set  $C_{\text{OPT}} = 1$  and we maximize  $C_{\text{ALG}}$ . Let  $V = \{A_0, A_1, O_0, O_1\}$  and  $V' = V \cup \{\mathcal{R}\}$ .

*maximize*  $C_{\text{ALG}}$

*subject to:*

$$C_{\text{ALG}} = \delta \cdot [A_0, \mathcal{R}] + [A_0, A_1] + \phi \cdot ([A_1, O_1] - [A_0, O_0])$$

$$C_{\text{OPT}} = 1$$

$$C_{\text{OPT}} = C_{\text{OPT}}^{\text{req}} + C_{\text{OPT}}^{\text{move}}$$

$$C_{\text{OPT}}^{\text{move}} \geq [O_0, O_1]$$

$$2 \cdot C_{\text{OPT}}^{\text{req}} + \delta \cdot C_{\text{OPT}}^{\text{move}} \geq \delta \cdot [O_0, \mathcal{R}] + \delta \cdot [O_1, \mathcal{R}]$$

$$f(A_1) \leq f(v)$$

$$\text{for all } v \in V$$

$$0 \leq [v_1, v_3] \leq [v_1, v_2] + [v_2, v_3]$$

$$\text{for all } v_1, v_2, v_3 \in V'$$

As  $\mathcal{R}$  is a set of requests, it does not necessarily correspond to a single point in the studied metric. Nevertheless, our notion of average distances (i.e.,  $[v_1, v_2]$ ) allows us to write the triangle inequalities for any pair of objects from set  $V \cup \{\mathcal{R}\}$ .

In the LP above,  $C_{\text{OPT}}^{\text{req}}$ ,  $C_{\text{OPT}}^{\text{move}}$  denote the cost of OPT for serving the request and the cost of OPT for migrating the file, respectively. The inequality  $2 \cdot C_{\text{OPT}}^{\text{req}} + \delta \cdot C_{\text{OPT}}^{\text{move}} \geq \delta \cdot [O_0, \mathcal{R}] + \delta \cdot [O_1, \mathcal{R}]$  is a counterpart of the relation guaranteed by Lemma 3.

For any choice of parameters  $\beta$ ,  $\delta$ , and  $\phi$ , the LP above finds an instance that maximizes the competitive ratio of ALG. Note that such instance is not necessarily a certificate that ALG indeed performs poorly: in particular, inequalities that lower-bound the cost of OPT might not be tight. However, the opposite is true: if the value of  $C_{\text{ALG}}$  returned by the LP is  $\xi$ , then for any possible instance the ratio is at most  $\xi$ .

Let  $c_0 = 1.841$  be the phase length of MTLM. Setting  $\delta = c_0$  and  $\beta = \phi = 1 + c_0$  yields that the optimal value of the LP is  $R_0 \approx 4.086$ , which can be interpreted as a numerical counterpart of the original analysis in [8]. To obtain a formal mathematical proof, one may take a dual solution to the LP. It gives the coefficients that multiplied by the corresponding LP inequalities and summed over all inequalities yield a proof that the amortized cost of MTLM in any phase is at most  $R_0$  times the cost of OPT. Summed over all phases, this implies that MTLM is  $R_0$ -competitive.

Among other advantages, this approach allows us to numerically find the instances that are tight for the current analysis (cf. Section 2 and Figure 1): linear and bipartite instances can be obtained this way.

## 4.2 LP analysis of DLM-like algorithms

Now we show how to adapt the LP from the previous section to analyze DLM-type algorithms. Recall that after  $1.75D$  requests, DLM evaluates the geometry of the so-far-received requests and decides whether to continue this phase or not. Although the final parameters of DLM are elegant numbers (multiplicities of  $1/4$ ), they were obtained by a tedious optimization process using the LP we present below. Furthermore, the LP below does not give us an explicit rule for continuing the phase; it only tells that DLM is successful either in a short or in a long phase.

Recall that in a phase, DLM considers three groups of consecutive  $\delta_i \cdot D$  requests:  $\mathcal{R}_1$ ,  $\mathcal{R}_2$ , and  $\mathcal{R}_3$ , where  $\delta_1$ ,  $\delta_2$ , and  $\delta_3$  are parameters of DLM. First, assume that DLM always processes three parts and afterwards it moves the file to a point  $A_3$  that minimizes the function

$$h(x) = [A_0, x] + \beta_1 \cdot [x, \mathcal{R}_1] + \beta_2 \cdot [x, \mathcal{R}_2] + \beta_3 \cdot [x, \mathcal{R}_3] ,$$

where  $\beta_i$  are parameters that we choose later. We denote the strategy of an optimal algorithm by OPTL (short for OPT-LONG). Let  $O_0^L$ ,  $O_1^L$ ,  $O_2^L$  and  $O_3^L$  denote the trajectory of OPTL ( $O_0^L$  is the initial position of the OPTL's file at the beginning of the phase, and  $O_i^L$  is its position right after the  $i$ -th part of the phase). Analogously to the previous section, we obtain the following LP.

*maximize*  $C_{\text{ALGL}}$

*subject to:*

$$\begin{aligned} C_{\text{ALGL}} &= [A_0, A_3] + \sum_{i=1,2,3} \delta_i \cdot [A_0, \mathcal{R}_i] + \phi \cdot ([A_3, O_3^L] - [A_0, O_0^L]) \\ C_{\text{OPTL}} &= 1 \\ C_{\text{OPTL}} &= \sum_{i=1,2,3} (C_{\text{OPTL}}^{\text{req}}(i) + C_{\text{OPTL}}^{\text{move}}(i)) \\ C_{\text{OPTL}}^{\text{move}}(i) &\geq [O_{i-1}^L, O_i^L] && \text{for } i = 1, 2, 3 \\ 2 \cdot C_{\text{OPTL}}^{\text{req}}(i) + \delta \cdot C_{\text{OPTL}}^{\text{move}}(i) &\geq \delta_i \cdot [O_{i-1}^L, \mathcal{R}_i] + \delta_i \cdot [O_i^L, \mathcal{R}_i] && \text{for } i = 1, 2, 3 \\ h(A_3) &\leq h(v) && \text{for all } v \in V \\ 0 \leq [v_1, v_3] &\leq [v_1, v_2] + [v_2, v_3] && \text{for all } v_1, v_2, v_3 \in V' \end{aligned}$$

This time  $V = \{A_0, A_3, O_0^L, O_1^L, O_2^L, O_3^L\}$  and  $V' = V \cup \{\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3\}$ .

We note that such parameterization alone does not improve the competitive ratio, i.e., for any choice of parameters  $\delta_i$  and  $\beta_i$ , the objective value of the LP above is at least  $R_0 \approx 4.086$ .

However, as stated in [Section 2.1](#), DLM verifies if after two parts it can migrate its file to a node  $A_2$  being the minimizer of the function

$$g(x) = [A_0, x] + \beta'_1 \cdot [x, \mathcal{R}_1] + \beta'_2 \cdot [x, \mathcal{R}_2] ,$$

where  $\beta'_i$  are parameters that we choose later.

In our analysis, we gave an explicit rule whether the migration to  $A_2$  should take place. However, for our LP-based approach, we follow a slightly different scheme. Namely, if the migration to  $A_2$  guarantees that the amortized cost in the short phase (the first two parts) is at most 4 times the cost of *any* strategy for the short phase, then DLM may move to  $A_2$  and we immediately achieve competitive ratio 4 on the short phase. Otherwise, we may add additional constraints to the LP, stating that the competitive ratio of an algorithm which moves to  $A_2$  is at least 4 (against any chosen strategy OPTS). Analogously to OPTL, the trajectory of OPTS is described by three points:  $O_0^S$ ,  $O_1^S$ , and  $O_2^S$ . This allows us to strengthen our LP by adding the following inequalities:

$$\begin{aligned} C_{\text{ALGS}} &= [A_0, A_2] + \sum_{i=1,2} \delta_i \cdot [A_0, \mathcal{R}_i] + \phi \cdot ([A_2, O_2^S] - [A_0, O_0^S]) \\ C_{\text{OPTS}} &= \sum_{i=1,2} (C_{\text{OPTS}}^{\text{req}}(i) + C_{\text{OPTS}}^{\text{move}}(i)) \\ C_{\text{OPTS}}^{\text{move}}(i) &\geq [O_{i-1}^S, O_i^S] && \text{for } i = 1, 2 \\ 2 \cdot C_{\text{OPTS}}^{\text{req}}(i) + \delta \cdot C_{\text{OPTS}}^{\text{move}}(i) &\geq \delta_i \cdot [O_{i-1}^S, \mathcal{R}_i] + \delta_i \cdot [O_i^S, \mathcal{R}_i] && \text{for } i = 1, 2 \\ g(A_2) &\leq g(v) && \text{for all } v \in V \\ C_{\text{ALGS}} &\geq 4 \cdot C_{\text{OPTS}} \end{aligned}$$

We also change  $V$  to  $\{A_0, A_3, O_0^L, O_1^L, O_2^L, O_3^L, O_0^S, O_1^S, O_2^S\}$ , both in new and in old inequalities.

When we choose  $\phi = 3$ , fix phase length parameters to be  $\delta_1 = 1$ ,  $\delta_2 = 0.75$ ,  $\delta_3 = 0.5$  and parameters for functions  $g$  and  $h$  to be  $\beta'_1 = 2$ ,  $\beta'_2 = 1$ ,  $\beta_1 = 1$ ,  $\beta_2 = 0.25$  and  $\beta_3 = 0.75$ , we obtain that the value of the above LP is 4. Again, this can be interpreted as a numerical argument that DLM is indeed a 4-competitive algorithm.

## 5 Conclusions

While in the last decade factor-revealing LPs became a standard tool for analysis of approximation algorithms, their application to online algorithms so far have been limited to online bipartite matching and its variants (see, e.g., [\[24, 20\]](#)) and for showing lower bounds [\[5\]](#). In this paper, we successfully used the factor-revealing LP to bound the competitive ratio of an algorithm for an online problem defined on an arbitrary metric space. We believe that similar approaches could yield improvements also for other online graph problems.

---

## References

- 1 Susanne Albers and Hisashi Koga. Page migration with limited local memory capacity. In *Proc. 4th Int. Workshop on Algorithms and Data Structures (WADS)*, pages 147–158, 1995.
- 2 Baruch Awerbuch, Yair Bartal, and Amos Fiat. Competitive distributed file allocation. In *Proc. 25th ACM Symp. on Theory of Computing (STOC)*, pages 164–173, 1993.
- 3 Baruch Awerbuch, Yair Bartal, and Amos Fiat. Heat & Dump: Competitive distributed paging. In *Proc. 34th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 22–31, 1993.

- 4 Baruch Awerbuch, Yair Bartal, and Amos Fiat. Distributed paging for general networks. *Journal of Algorithms*, 28(1):67–104, 1998.
- 5 Yossi Azar, Ilan Reuven Cohen, and Alan Roytman. Online lower bounds via duality. In *Proc. 28th ACM-SIAM Symp. on Discrete Algorithms (SODA)*, pages 1038–1050, 2017.
- 6 Yair Bartal. *Competitive Analysis of Distributed On-line Problems — Distributed Paging*. PhD thesis, Tel-Aviv University, 1995.
- 7 Yair Bartal. Distributed paging. In *Dagstuhl Workshop on On-line Algorithms*, pages 97–117, 1996.
- 8 Yair Bartal, Moses Charikar, and Piotr Indyk. On page migration and other relaxed task systems. *Theoretical Computer Science*, 268(1):43–66, 2001. Also appeared in *Proc. of the 8th SODA*, pages 43–52, 1997.
- 9 Yair Bartal, Amos Fiat, and Yuval Rabani. Competitive algorithms for distributed data management. *Journal of Computer and System Sciences*, 51(3):341–358, 1995.
- 10 Shai Ben-David, Allan Borodin, Richard M. Karp, Gabor Tardos, and Avi Wigderson. On the power of randomization in online algorithms. *Algorithmica*, 11(1):2–14, 1994.
- 11 Marcin Bienkowski. Migrating and replicating data in networks. *Computer Science — Research and Development*, 27(3):169–179, 2012.
- 12 Marcin Bienkowski, Jaroslaw Byrka, Mirosław Korzeniowski, and Friedhelm Meyer auf der Heide. Optimal algorithms for page migration in dynamic networks. *Journal of Discrete Algorithms*, 7(4):545–569, 2009.
- 13 David L. Black and Daniel D. Sleator. Competitive algorithms for replication and migration problems. Technical Report CMU-CS-89-201, Department of Computer Science, Carnegie-Mellon University, 1989.
- 14 Allan Borodin and Ran El-Yaniv. *Online Computation and Competitive Analysis*. Cambridge University Press, 1998.
- 15 Marek Chrobak, Lawrence L. Larmore, Nick Reingold, and Jeffery Westbrook. Page migration algorithms using work functions. *Journal of Algorithms*, 24(1):124–157, 1997.
- 16 Bezael Gavish and Olivia R. Liu Sheng. Dynamic file migration in distributed computer systems. *Communications of the ACM*, 33(2):177–189, 1990.
- 17 Makoto Imase and Bernard M. Waxman. Dynamic Steiner tree problem. *SIAM Journal on Discrete Mathematics*, 4(3):369–384, 1991.
- 18 Carsten Lund, Nick Reingold, Jeffery Westbrook, and Dicky C. K. Yan. Competitive on-line algorithms for distributed data management. *SIAM Journal on Computing*, 28(3):1086–1111, 1999.
- 19 Bruce M. Maggs, Friedhelm Meyer auf der Heide, Berthold Vöcking, and Matthias Westermann. Exploiting locality for data management in systems of limited bandwidth. In *Proc. 38th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 284–293, 1997.
- 20 Mohammad Mahdian and Qiqi Yan. Online bipartite matching with random arrivals: an approach based on strongly factor-revealing LPs. In *Proc. 43rd ACM Symp. on Theory of Computing (STOC)*, pages 597–606, 2011.
- 21 Akira Matsubayashi. Uniform page migration on general networks. *International Journal of Pure and Applied Mathematics*, 42(2):161–168, 2008.
- 22 Akira Matsubayashi. A  $3+\Omega(1)$  lower bound for page migration. In *Proc. 3rd Int. Symp. on Computing and Networking (CANDAR)*, pages 314–320, 2015.
- 23 Akira Matsubayashi. Asymptotically optimal online page migration on three points. *Algorithmica*, 71(4):1035–1064, 2015.
- 24 Aranyak Mehta, Amin Saberi, Umesh V. Vazirani, and Vijay V. Vazirani. Adwords and generalized online matching. *Journal of the ACM*, 54(5), 2007.

- 25 Friedhelm Meyer auf der Heide, Berthold Vöcking, and Matthias Westermann. Provably good and practical strategies for non-uniform data management in networks. In *Proc. 7th European Symp. on Algorithms (ESA)*, pages 89–100, 1999.
- 26 Jeffery Westbrook. Randomized algorithms for the multiprocessor page migration. *SIAM Journal on Computing*, 23:951–965, 1994.

## A Omitted Proofs

► **Lemma 3.** *Let  $\mathcal{R}$  be set subsequence of at most  $2D$  consecutive requests from the input issued at steps  $t+1, t+2, \dots, t+|\mathcal{R}|$ . Then,  $4 \cdot C_{\text{OPT}}(\mathcal{R}) \geq (2|\mathcal{R}|/D) \cdot [\text{OP}_t, \mathcal{R}, \text{OP}_{t+|\mathcal{R}|}] + (4 - 2|\mathcal{R}|/D) \cdot [\text{OP}_t, \text{OP}_{t+|\mathcal{R}|}]$ .*

**Proof.** For simplicity of notation, we assume that  $t = 0$ . In these terms,  $\mathcal{R}$  corresponds to requests  $r_1, r_2, \dots, r_{|\mathcal{R}|}$  issued at the consecutive steps. By the triangle inequality, for any  $y \in \{0, \dots, R\}$ , it holds that

$$\frac{|\mathcal{R}|}{D} \cdot [\text{OP}_y, \mathcal{R}] = \sum_{i=1}^{|\mathcal{R}|} d(\text{OP}_y, r_i) \leq \sum_{i=1}^{|\mathcal{R}|} d(\text{OP}_y, \text{OP}_{i-1}) + \sum_{i=1}^{|\mathcal{R}|} d(\text{OP}_{i-1}, r_i) .$$

Hence,

$$\begin{aligned} \frac{|\mathcal{R}|}{D} \cdot [\text{OP}_0, \mathcal{R}, \text{OP}_{|\mathcal{R}|}] &\leq \sum_{i=1}^{|\mathcal{R}|} \left( d(\text{OP}_0, \text{OP}_{i-1}) + d(\text{OP}_{|\mathcal{R}|}, \text{OP}_{i-1}) \right) + 2 \sum_{i=1}^{|\mathcal{R}|} d(\text{OP}_{i-1}, r_i) \\ &\leq \sum_{i=1}^{|\mathcal{R}|} \sum_{j=1}^{|\mathcal{R}|} d(\text{OP}_{j-1}, \text{OP}_j) + 2 \sum_{i=1}^{|\mathcal{R}|} d(\text{OP}_{i-1}, r_i) \\ &= |\mathcal{R}| \sum_{j=1}^{|\mathcal{R}|} d(\text{OP}_{j-1}, \text{OP}_j) + 2 \sum_{i=1}^{|\mathcal{R}|} d(\text{OP}_{i-1}, r_i) . \end{aligned}$$

Finally,

$$\begin{aligned} &\frac{2|\mathcal{R}|}{D} \cdot [\text{OP}_0, \mathcal{R}, \text{OP}_{|\mathcal{R}|}] + \left( 4 - \frac{2|\mathcal{R}|}{D} \right) \cdot [\text{OP}_0, \text{OP}_{|\mathcal{R}|}] \\ &\leq 4 \sum_{i=1}^{|\mathcal{R}|} d(\text{OP}_{i-1}, r_i) + 2|\mathcal{R}| \sum_{i=1}^{|\mathcal{R}|} d(\text{OP}_{i-1}, \text{OP}_i) + \left( 4 - \frac{2|\mathcal{R}|}{D} \right) \cdot D \cdot \sum_{i=1}^{|\mathcal{R}|} d(\text{OP}_{i-1}, \text{OP}_i) \\ &= 4 \sum_{i=1}^{|\mathcal{R}|} d(\text{OP}_{i-1}, r_i) + 4D \sum_{i=1}^{|\mathcal{R}|} d(\text{OP}_{i-1}, \text{OP}_i) \\ &= 4 \cdot C_{\text{OPT}}(\mathcal{R}) . \end{aligned} \quad \blacktriangleleft$$

## B Lower Bound for Phase-Based Algorithms

In this section, we show that, under some additional assumptions, no fixed-phase algorithm can beat the competitive ratio  $R_0 \approx 4.086$  achieved by MTLM (see [Section 1.2](#)), where  $R_0$  is the largest (real) root of the equation

$$R^3 - 5R^2 + 3R + 3 = 0 . \quad (11)$$

Let  $v_{\text{OPT}}$  and  $v_{\text{ALG}}$  be the positions of files of OPT and ALG, respectively. ALG and OPT start at the same point of the metric. Recall that a fixed-phase algorithm chooses phase

length  $c \cdot D$  and after every  $c \cdot D$  requests it makes a migration decision solely on the basis of its current position and the last  $c \cdot D$  requests. In particular, it cannot store the history of past requests beyond the window consisting of the last  $c \cdot D$  requests. Bartal et al. [8] showed that no fixed-phase algorithm can achieve competitive ratio better than 3.847 (for  $D$  tending to infinity).

We present our lower bound in a model that gives an additional power to the adversary. Let  $f$  denote the distance between  $v_{\text{ALG}}$  and  $v_{\text{OPT}}$  at the end of a phase. Then, at the beginning of the next phase  $P$ , the adversary removes the existing graph and creates a completely new one in which it chooses a new position for  $v_{\text{ALG}}$ . It creates a sequence of requests constituting phase  $P$  and runs ALG on  $P$ . Finally, it chooses a strategy for OPT for  $P$ , with the restriction that the initial distance between OPT and ALG files is exactly  $f$ . We call this setting *dynamic graph model*.

## B.1 Using known lower bound for short phases

The lower bound given for fixed-phases algorithms by Bartal et al. [8] is already sufficient to show the desired lower bound for shorter phase lengths. (It can also be used for very long phases, but we do not use this property.)

► **Lemma 4.** *Let  $c_T = 2(R_0 + 1)/(R_0^2 - 2R_0 - 1) \approx 1.352$ . No fixed-phase algorithm using phase lengths  $c \cdot D$  with  $c \leq c_T$  can achieve competitive ratio lower than  $R_0$ .*

**Proof.** Theorem 3.2 of [8] states that no algorithm using phases of length  $c \cdot D$  can have competitive ratio smaller than

$$L(c) = \inf_{a \in (0,1)} \max \left\{ \frac{a}{1-a}, \frac{c+2}{ca} + 1, c \cdot (a+1) + 1 \right\}. \quad (12)$$

Theorem 3.2 of [8] also shows that  $L(c) \geq 3.847$  for any  $c$ . However, we may also analyze this bound restricting phase lengths. In particular, we claim that when  $c \leq c_T := 2(R_0 + 1)/(R_0^2 - 2R_0 - 1)$ , then  $L(c) \geq R_0$ . To see this, consider two cases. When  $a > R_0/(1 + R_0)$ , the first term of (12),  $a/(1-a)$ , is already greater than  $R_0$ . Otherwise,  $a < R_0/(1 + R_0)$  and for  $c \leq c_T$ , the second term of (12),  $(c+2)/(ca) + 1$ , is at least  $R_0$ . ◀

## B.2 States

As mentioned in the informal introduction given in Section 3, our adversarial construction will consist of multiple epochs, each consisting of multiple phases. In each phase, the adversary will employ one of three possible plays: *linear*, *bipartite* or *finishing*, which were sketched in Section 3 and will be formally introduced in the subsequent sections.

Our construction will be parameterized with integers  $L$  and  $k$ ; the latter is a parameter used in the bipartite play. We define

$$\varepsilon = \max \left\{ \sum_{i=L}^{\infty} (2\alpha)^i, \frac{4R_0}{k+4} \right\} = \max \left\{ \frac{(2\alpha)^L}{1-2\alpha}, \frac{4R_0}{k+4} \right\},$$

where  $\alpha = 1/(R_0 - 1)$ . Note that  $\varepsilon$  tends to zero with increasing  $L$  and  $k$ . Our goal is to show that for any epoch  $E$  it holds that  $C_{\text{ALG}}(E) \geq (R_0 - \varepsilon) \cdot C_{\text{OPT}}(E)$ . As  $\varepsilon$  can be made arbitrarily small, this will imply the lower bound of  $R_0$ .

The relation  $C_{\text{ALG}}(M) \geq (R_0 - \varepsilon) \cdot C_{\text{OPT}}(M)$  does not hold for each adversarial play  $M$ . Nonetheless, we want to measure the amount  $C_{\text{ALG}}(M) - (R_0 - \varepsilon) \cdot C_{\text{OPT}}(M)$  for each



play  $M$ ; we call this amount *play gain*. For a single play  $M$ , different behaviors of ALG can result in different play gains and final values of  $d(v_{\text{ALG}}, v_{\text{OPT}})$ .

First, we define possible *states*. A state is defined between phases and depends on the distance between  $v_{\text{ALG}}$  and  $v_{\text{OPT}}$ .

1. State  $S$ :  $v_{\text{ALG}} = v_{\text{OPT}}$ .
2. State  $A_\ell$  for  $\ell \in \{0, \dots, L\}$ :  $d(v_{\text{ALG}}, v_{\text{OPT}}) = (2\alpha)^\ell$ , where  $\alpha = 1/(R_0 - 1)$ .
3. State  $F$ : all remaining distances between  $v_{\text{ALG}}$  and  $v_{\text{OPT}}$ .

In the following sections, we define adversarial plays: for each state there will be one play that can start at this state. For each play, we will then characterize possible outcomes: play gains and the resulting states. Finally, we analyze the total gain on any sequence of plays and we show that it can be lower-bounded by a constant. This will imply [Theorem 2](#).

### B.3 Starting in state $S$ : linear play

► **Lemma 5.** *Assume a phase starts in state  $S$ . Then, the adversary may employ a single-phase (linear) play, whose gain is at least  $-\sum_{i=0}^{L-1} (2\alpha)^i \cdot D$  and which always ends in state  $A_0$ .*

**Proof.** At the beginning of a linear play,  $v_{\text{ALG}} = v_{\text{OPT}}$ . The created graph consists of two nodes,  $a = v_{\text{ALG}} = v_{\text{OPT}}$  and  $b$ , connected with an edge of length 1. Let  $t = 1 + 1/R_0 \approx 1.245$ . Observe that  $t < c$  for all considered lengths  $c \cdot D$  of a phase. The play consists of a single phase  $P$ , whose first  $(c - t) \cdot D$  requests are given at  $a$  and the following  $t \cdot D$  requests are given at  $b$ .

Note that ALG pays 1 for each of the last  $t$  requests. We consider two cases depending on the possible action of ALG at the end of  $P$ .

1. ALG migrates the file to  $b$ . In this case  $C_{\text{ALG}}(P) = (t + 1) \cdot D$ . OPT then chooses to keep its file at  $a$  throughout  $P$  paying  $t \cdot D$ . Then,

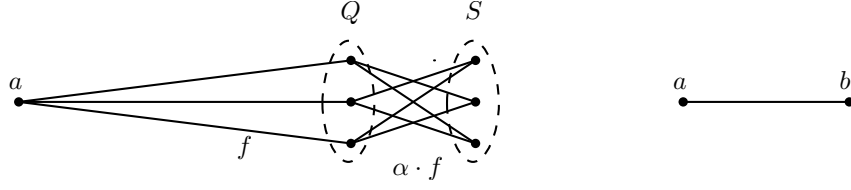
$$C_{\text{ALG}}(P) - R_0 \cdot C_{\text{OPT}}(P) = (t + 1 - R_0 \cdot t) \cdot D = (1/R_0 + 1 - R_0) \cdot D .$$

2. ALG keeps the file at  $a$ . In this case  $C_{\text{ALG}}(P) = t \cdot D$ . OPT then remains at  $a$  for the first  $c - t$  requests, migrates its file to  $b$ , and keeps it there till the end of  $P$ . Altogether,  $C_{\text{OPT}}(P) = D$ . Then,

$$C_{\text{ALG}}(P) - R_0 \cdot C_{\text{OPT}}(P) = (t - R_0 \cdot 1) \cdot D = (1/R_0 + 1 - R_0) \cdot D .$$

In both cases, the resulting state is  $A_0$ . Using the definition of  $R_0$  (see (11)), it can be verified that  $R_0 - 1 - 1/R_0 = (R_0 - 1)/(R_0 - 3)$ . By the definition of  $\alpha$ , this is equal to  $1/(1 - 2\alpha)$ . Therefore, using  $C_{\text{OPT}}(P) \geq D$ , we obtain that the play gain is

$$\begin{aligned} C_{\text{ALG}}(P) - (R_0 - \varepsilon) \cdot C_{\text{OPT}}(P) &= \varepsilon \cdot C_{\text{OPT}}(P) - (R_0 - 1 - 1/R_0) \cdot D \\ &\geq \left( \varepsilon - \frac{1}{1 - 2\alpha} \right) \cdot D = \left( \varepsilon - \sum_{i=0}^{\infty} (2\alpha)^i \right) \cdot D \\ &\geq - \sum_{i=0}^{L-1} (2\alpha)^i \cdot D . \end{aligned} \quad \blacktriangleleft$$



■ **Figure 2** An example of graph used in the bipartite play for  $k = 3$  (left) and a graph used in the linear play (right). Note that they correspond to tight examples for the performance of the algorithm MTLM.

#### B.4 Starting in state $A_\ell$ : bipartite play

► **Lemma 6.** Assume a phase starts in state  $A_\ell$  for  $\ell \in \{0, \dots, L-1\}$ . Then, the adversary may employ a single-phase (bipartite) play, such that one of the following conditions hold:

1. the resulting state is  $A_\ell$  and the play gain is at least zero;
2. the resulting state is  $A_{\ell+1}$  and the play gain is at least  $(2\alpha)^\ell \cdot D$ ;
3. the resulting state is  $F$  with  $v_{\text{ALG}} - v_{\text{OPT}}$  distance equal to  $3\alpha \cdot (2\alpha)^\ell$  and the play gain is at least  $(1 + \alpha) \cdot (2\alpha)^\ell \cdot D$ .

**Proof.** Let  $f = (2\alpha)^\ell$  denote the initial value of  $d(v_{\text{ALG}}, v_{\text{OPT}})$ . Let  $a$  denote the initial position of ALG's file.

*Phase construction.* The construction of this play will be parameterized by an integer  $k \geq 3$ . The graph created by the adversary will be bipartite and consists of the following three parts:  $\{a\}$ , set  $Q$ , and set  $S$ , where  $|Q| = |S| = k$ , see Figure 2. As allowed in the dynamic graph model, the exact initial position of  $v_{\text{OPT}}$  will be determined later based on the behavior of ALG; in any case it will be initially in set  $Q$ . Node  $a$  is connected with all nodes from  $Q$  with an edge of length  $f$ . The connections between  $Q$  and  $S$  constitute an almost complete bipartite graph, whose edges are of length  $\alpha \cdot f$ . Namely, we number all nodes from  $Q$  and  $S$  as  $q_1, q_2, \dots, q_k$  and  $s_1, s_2, \dots, s_k$ , respectively, and we connect  $q_i$  with  $s_j$  if and only if  $i \neq j$ . An example for  $k = 3$  is given in Figure 2. As  $k \geq 3$ , any pair of nodes from  $S$  share a common neighbor from  $Q$  and hence the distance between them is exactly  $2\alpha \cdot f$ .

All the requests are given at nodes from  $S$  in a round-robin fashion (the adversary fixes an arbitrary ordering of nodes from  $S$  first). As  $D$  is sufficiently large, each node of  $S$  issues at least one request.

*Cost analysis.* In either case OPT remains at one node from  $Q$  for the whole phase  $P$ . It pays  $\alpha \cdot f$  for any request at  $k - 1$  neighboring nodes from  $Q$  and  $3\alpha \cdot f$  for any request at the only non-incident node from  $Q$ . As requests are given in a round-robin fashion, the number of requests at that non-incident node is  $m \leq \lceil cD/k \rceil \leq 2cD/k$ , and the total cost of OPT is

$$\begin{aligned} C_{\text{OPT}}(P) &= \alpha \cdot f \cdot (cD - m) + 3\alpha \cdot f \cdot m \\ &= \alpha \cdot f \cdot (cD + 2m) \\ &\leq (1 + 4/k) \cdot \alpha \cdot f \cdot cD . \end{aligned}$$

By the definition of  $\varepsilon$ , it holds that  $(R_0 - \varepsilon) \cdot (1 + 4/k) = R_0 + (4R_0/k - \varepsilon \cdot (1 + 4/k)) \leq R_0$ . Furthermore, we split the cost of ALG on  $P$  into the cost of serving the requests,  $C_{\text{ALG}}^R(P)$ ,

and the migration cost  $C_{\text{ALG}}^M(P)$ . The former is exactly  $C_{\text{ALG}}^R(P) = (1 + \alpha) \cdot f \cdot cD$ . Therefore,

$$\begin{aligned} C_{\text{ALG}}(P) - (R_0 - \varepsilon) \cdot C_{\text{OPT}}(P) &= C_{\text{ALG}}^M(P) + C_{\text{ALG}}^R(P) - (R_0 - \varepsilon) \cdot C_{\text{OPT}}(P) \\ &\geq C_{\text{ALG}}^M(P) + (1 + \alpha) \cdot f \cdot cD - R_0 \cdot \alpha \cdot f \cdot cD \\ &= C_{\text{ALG}}^M(P) , \end{aligned}$$

where the last equality follows as  $R_0 \cdot \alpha = 1 + \alpha$  by the definition of  $\alpha$ . Hence, for lower-bounding the play gain, it is sufficient to lower-bound  $C_{\text{ALG}}^M(P)$ . We consider several possible migration options for ALG on the bipartite play.

1. ALG remains at  $a$ . In this case  $C_{\text{ALG}}^M(P) = 0$ , and the resulting state is still  $A_\ell$ .
2. ALG migrates the file to a node  $q \in Q$ , paying  $f \cdot D = (2\alpha)^\ell \cdot D$  for the migration. The adversary chooses its original position to be any node from  $Q$  different from  $q$ . Therefore, the final distance between ALG and OPT files is exactly  $2\alpha \cdot f$ . The resulting state is  $A_{\ell+1}$  and the play gain is at least  $C_{\text{ALG}}^M(P) = (2\alpha)^\ell \cdot D$ .
3. ALG migrates the file to a node  $s \in S$ . The adversary chooses its original position to be (the only) node from  $Q$  not directly connected to  $s$ . The cost of migration is  $(1 + \alpha) \cdot f \cdot D$  and the resulting distance between ALG and OPT files is then  $3\alpha \cdot f$ , i.e., the play ends in state  $F$ . ◀

## B.5 Starting in states F and $A_L$ : finishing play

► **Lemma 7.** *Assume a phase starts in state  $A_L$  or  $F$  and the initial distance between  $v_{\text{ALG}}$  and  $v_{\text{OPT}}$  is  $f$ . Then, the adversary may employ a (finishing) play, whose gain is at least  $(c + 1) \cdot f \cdot D$  and which always ends in state  $S$ .*

**Proof.** In this case the adversary uses a finishing play  $M$ . This is the only play that may consist of more than one phase. It is played on two nodes  $v_{\text{ALG}}$  and  $v_{\text{OPT}}$ , connected by an edge of length  $f$ . In a phase of this play, OPT never moves and all requests are issued at  $v_{\text{OPT}}$ . If at the end of the phase ALG does not migrate the file to  $v_{\text{OPT}}$ , the adversary repeats the phase.

The cost of OPT in any such phase is 0. Hence, any competitive algorithm has to finally migrate to  $v_{\text{OPT}}$ , possibly over a sequence of multiple phases, i.e., the final state is always of type  $S$ . In the first phase of  $M$ , ALG pays at least  $c \cdot D \cdot f$  for the requests. Furthermore, within  $M$ , ALG migrates the file along the distance of at least  $f$ , paying  $f \cdot D$ . The play gain is then

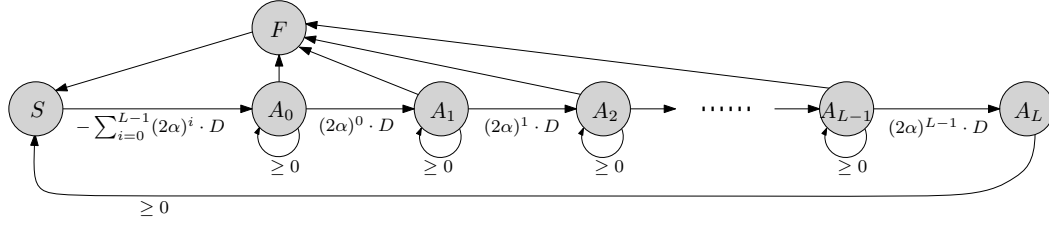
$$C_{\text{ALG}}(M) - (R_0 - \varepsilon) \cdot C_{\text{OPT}}(M) \geq C_{\text{ALG}}(M) \geq (c + 1) \cdot f \cdot D . \quad \blacktriangleleft$$

## B.6 Combining all plays

In Figure 3, we summarized the possible transitions between states, as described in Lemma 5, Lemma 6, and Lemma 7. We may now use this state graph to show the desired lower bound.

► **Lemma 8.** *Fix any algorithm ALG and a input (sub)sequence  $\mathcal{I}$  consisting of some number of plays, such that ALG starts and ends  $\mathcal{I}$  at state  $S$ . Then, the total gain on all plays from  $\mathcal{I}$  is non-negative.*

**Proof.** For a transition between two states  $Q_1$  and  $Q_2$ , we denote its gain by  $T(Q_1, Q_2)$ . By Lemma 6,  $T(A_\ell, A_\ell) \geq 0$  for any  $\ell \in \{0, \dots, L - 1\}$ . Thus, we may ignore loops at  $A_\ell$  when computing total plays' gain. There are two possibilities:



■ **Figure 3** Plays as transitions between states. Note that state  $F$  corresponds to many possible configurations with different distances between  $v_{\text{ALG}}$  and  $v_{\text{OPT}}$ . The gains for all plays that avoid state  $F$  are given at the corresponding edges.

1. The game starts at state  $S$ , goes through states  $A_0, A_1, \dots, A_{L-1}, A_L$  (possibly looping at some  $A_\ell$ 's), and returns to  $S$ . The total plays' gain is then

$$T(S, A_0) + \sum_{\ell=0}^{L-1} T(A_\ell, A_{\ell+1}) + T(A_L, S) \geq - \sum_{i=0}^{L-1} (2\alpha)^i \cdot D + \sum_{\ell=0}^{L-1} (2\alpha)^\ell \cdot D \geq 0 .$$

2. The game starts at state  $S$ , goes through states  $A_0, A_1, \dots, A_m$  (for  $m < L$ , possibly looping at some  $A_\ell$ 's), then to state  $F$ , and returns to  $S$ . Note that the distance between  $v_{\text{ALG}}$  and  $v_{\text{OPT}}$  at state  $A_m$  is  $(2\alpha)^m$  and therefore this distance in state  $F$  is  $3\alpha \cdot (2\alpha)^m$ . The total plays' gain is then

$$\begin{aligned} & T(S, A_0) + \sum_{\ell=0}^{m-1} T(A_\ell, A_{\ell+1}) + T(A_m, F) + T(F, S) \\ & \geq - \sum_{i=0}^{L-1} (2\alpha)^i \cdot D + \sum_{\ell=0}^{m-1} (2\alpha)^\ell \cdot D + (1 + \alpha)(2\alpha)^m \cdot D + (c + 1) \cdot 3\alpha \cdot (2\alpha)^m \cdot D \\ & = (2\alpha)^m \cdot D \cdot \left( (1 + \alpha) + (c + 1) \cdot 3\alpha - \sum_{i=0}^{L-m-1} (2\alpha)^i \right) \\ & \geq (2\alpha)^m \cdot D \cdot \left( (1 + \alpha) + (c + 1) \cdot 3\alpha - \frac{1}{1 - 2\alpha} \right) \\ & \geq 0 . \end{aligned}$$

The last inequality can be verified numerically: for  $\alpha = 1/(R_0 - 1) \approx 0.324$  and  $c \geq c_T = 2(R_0 + 1)/(R^2 - 2R_0 - 1) \approx 1.352$ , it holds that  $(1 + \alpha) + (c + 1) \cdot 3\alpha - 1/(1 - 2\alpha) > 0.768 > 0$ . ◀

► **Theorem 2.** Fix any algorithm  $\text{ALG}$  that operates in phases of fixed length. Assume that between the phases, the adversary can arbitrarily modify the graph while keeping the distance between the files of  $\text{OPT}$  and  $\text{ALG}$  unchanged. Then, the competitive ratio of  $\text{ALG}$  is at least  $R_0$  (for  $D$  tending to infinity), where  $R_0 \approx 4.086$  is the competitive ratio of algorithm  $\text{MTLM}$ .

**Proof.** We consider a sequence of state transitions that starts from state  $S$ . Assume that we run  $\text{ALG}$  infinitely and we observe its behavior.

If on an infinite sequence,  $\text{ALG}$  always returns to state  $S$ , then for an actual input we take any prefix  $\mathcal{I}$  of this infinite sequence ending at state  $S$ . By Lemma 8,  $C_{\text{ALG}}(\mathcal{I}) \geq (R_0 - \varepsilon) \cdot C_{\text{OPT}}(\mathcal{I})$  and  $C_{\text{ALG}}(\mathcal{I})$  can be made arbitrarily large by taking sufficiently long  $\mathcal{I}$ . As  $\varepsilon$  can be made arbitrarily small, the theorem follows.

Otherwise, ALG run on this infinite sequence returns to  $S$  fixed number of times, then reaches state  $A_\ell$  (for  $\ell \in \{0, \dots, L-1\}$ ), and then loops infinitely at  $A_\ell$ . If we stop the infinite sequence after some number of such loops, it defines an actual finite sequence  $\mathcal{I}$ , consisting of  $1 + 2(\ell + 1)$  parts:  $\mathcal{I}_{\text{init}}, \mathcal{I}_{\text{trans}}^0, \mathcal{I}_{\text{loop}}^0, \mathcal{I}_{\text{trans}}^1, \mathcal{I}_{\text{loop}}^1, \mathcal{I}_{\text{trans}}^2, \mathcal{I}_{\text{loop}}^2, \dots, \mathcal{I}_{\text{trans}}^\ell, \mathcal{I}_{\text{loop}}^\ell$ , where

1.  $\mathcal{I}_{\text{init}}$  starts and ends at  $S$ ;
2.  $\mathcal{I}_{\text{trans}}^i$  starts at  $A_{i-1}$  and ends at  $A_i$  (where by  $A_{-1}$  we understand  $S$ );
3.  $\mathcal{I}_{\text{loop}}^i$  loops at  $A_i$ .

Some parts  $\mathcal{I}_{\text{loop}}^i$  may be non-existing (empty sub-sequences). As in the previous case, the total plays' gain on  $\mathcal{I}_{\text{init}}$  is non-negative. By [Lemma 6](#), the same holds also for any part  $\mathcal{I}_{\text{loop}}^i$ . Therefore,

$$\begin{aligned}
 C_{\text{ALG}}(\mathcal{I}) &= C_{\text{ALG}}(\mathcal{I}_{\text{init}}) + \sum_{i=0}^{\ell} (C_{\text{ALG}}(\mathcal{I}_{\text{trans}}^i) + C_{\text{ALG}}(\mathcal{I}_{\text{loop}}^i)) \\
 &\geq C_{\text{ALG}}(\mathcal{I}_{\text{init}}) + \sum_{i=0}^{\ell} C_{\text{ALG}}(\mathcal{I}_{\text{loop}}^i) \\
 &\geq (R_0 - \varepsilon) \cdot \left( C_{\text{OPT}}(\mathcal{I}_{\text{init}}) + \sum_{i=0}^{\ell} C_{\text{OPT}}(\mathcal{I}_{\text{loop}}^i) \right) \\
 &= (R_0 - \varepsilon) \cdot \left( C_{\text{OPT}}(\mathcal{I}) - \sum_{i=0}^{\ell} C_{\text{OPT}}(\mathcal{I}_{\text{trans}}^i) \right).
 \end{aligned}$$

On the other hand, the total cost of OPT on all parts  $\mathcal{I}_{\text{trans}}^i$  can be upper-bounded by a universal constant  $\gamma$ , independent of  $\mathcal{I}$ . Thus,  $C_{\text{ALG}}(\mathcal{I}) \geq (R_0 - \varepsilon) \cdot C_{\text{OPT}}(\mathcal{I}) - (R_0 - \varepsilon) \cdot \gamma$ .

As  $C_{\text{ALG}}(\mathcal{I})$  can be made arbitrarily large (the cost of any loop from  $A_\ell$  to  $A_\ell$  is universally lower-bounded), the term  $(R_0 - \varepsilon) \cdot \gamma$  becomes negligible on the long run. Again, as  $\varepsilon$  can be made arbitrarily small, the theorem follows.  $\blacktriangleleft$